

Sso Login

SSO Login The `<aava-sso-login>` component is a comprehensive authentication interface that provides both traditional username/password login and Single Sign-On (SSO) capabilities. It features form validation, multiple size variants, loading states, and comprehensive event handling for building secure authentication systems, user portals, and enterprise login interfaces.

How to use

- import { AavaSSOLoginComponent } from "@aava/play-core";
- Usage ■ A complete SSO login form with username/password fields, keep signed in option, and SSO login button.

Features

- Authentication Methods ■ Traditional Login : Username/password authentication with form validation
- SSO Integration : Single Sign-On button for enterprise authentication
- Form Validation : Real-time validation with error messages
- Password Visibility : Toggle password visibility for better UX
- Keep Signed In : Remember user session option
- User Experience ■ Multiple Variants : Five size variants (xs, sm, md, lg, xl) for different contexts
- Loading States : Visual feedback during authentication processes
- Error Handling : Clear error messages and validation feedback
- Responsive Design : Adapts to different screen sizes and devices
- Accessibility : Full keyboard navigation and screen reader support
- Form Management ■ Reactive Forms : Built with Angular Reactive Forms for robust form handling
- Validation : Required field validation with custom error messages
- State Management : Comprehensive form state tracking and management
- Event Handling : Detailed event system for all user interactions
- Form Submission : Proper form submission with validation checks
- Configuration ■ Flexible Sizing : Configurable component sizes with automatic child component scaling
- Custom Styling : CSS custom properties for theming and customization
- Event System : Comprehensive event emission for integration
- Disabled States : Support for disabled states during processing
- API Reference ■ Inputs ■ Property Type
- Description loading boolean false Whether the component is in a loading state
- disabled boolean false Whether the component is disabled
- errorMessage string "Custom error message to display
- variant SSOLoginVariant 'md' Size variant (xs, sm, md, lg, xl)
- Outputs ■ Property Type
- Description login EventEmitter<SSOLoginCredentials> Emitted when login form is submitted
- ssoLogin EventEmitter<void> Emitted when SSO login button is clicked
- forgotPassword EventEmitter<string> Emitted when forgot password is clicked
- troubleSignin EventEmitter<void> Emitted when trouble signing in is clicked
- loginEvent EventEmitter<SSOLoginEvent> Emitted for all login-related events
- Methods** ■ Method Description
- onSubmit() Handles form submission and validation
- onSSOLogin() Handles SSO login button click
- onForgotPassword() Handles forgot password link click
- onTroubleSigningIn() Handles trouble signing in link click
- togglePasswordVisibility() Toggles password field visibility
- getContainerClasses() Returns CSS classes for container styling
- getButtonSize() Returns appropriate button size for variant
- getCheckboxSize() Returns appropriate checkbox size for variant
- getHyperlinkSize() Returns appropriate link size for variant
- Interfaces** ■ SSOLoginCredentials ■ interface
- SSOLoginCredentials { username : string ; // User's username or email password : string ; // }

User's password keepSignedIn ? : boolean ; // Whether to keep user signed in } SSOLoginEvent ■ interface SSOLoginEvent { type : "login" | "sso-login" | "forgot-password" | "trouble-signin" ; data : unknown ; credentials ? : SSOLoginCredentials ; } SSOLoginVariant ■ type SSOLoginVariant = "xs" | "sm" | "md" | "lg" | "xl" ; CSS Classes ■ The component provides several CSS classes for styling: Class Name Description .sso-login-container Main container wrapper .sso-login-container--xs Extra small variant styling .sso-login-container--sm Small variant styling .sso-login-container--md Medium variant styling (default) .sso-login-container--lg Large variant styling .sso-login-container--xl Extra large variant styling .login-header Header section container .login-title Main title styling .login-subtitle Subtitle styling .login-form Form container .form-options Options row (checkbox + link) .divider-section Divider with "or" text .divider-text "or" text styling .sso-section SSO button section .sso-highlight SSO text highlighting .trouble-signin Trouble signing in section .trouble-text Trouble text styling CSS Custom Properties ■ The component uses CSS custom properties for theming: Property Description --global-spacing-5 Container padding --global-spacing-6 Gap between sections --global-spacing-3 Header element spacing --global-radius-lg Container border radius --color-surface-subtle-hover Container background color --color-text-primary Primary text color --font-family-heading Heading font family --font-family-body Body text font family --global-font-size-xl Title font size --global-font-size-md Subtitle font size --global-font-size-sm Body text font size --global-font-size-xs Small text font size --global-font-weight-bold Bold font weight --global-font-weight-regular Regular font weight --global-line-height-loose Loose line height --global-line-height-tight Tight line height Best Practices ■ Authentication Security ■ Input Validation : Always validate inputs on both client and server side Password Security : Implement strong password requirements Rate Limiting : Add rate limiting for login attempts HTTPS Only : Ensure authentication occurs over secure connections Session Management : Implement proper session timeout and management User Experience ■ Clear Feedback : Provide immediate feedback for all user actions Error Messages : Use clear, actionable error messages Loading States : Show loading indicators during authentication Accessibility : Ensure full keyboard and screen reader support Mobile Optimization : Optimize for mobile devices and touch interaction Form Handling ■ Validation Timing : Validate on blur and submit for better UX Error Display : Show errors near the relevant fields Form State : Maintain form state during validation Submission : Prevent multiple form submissions Reset Handling : Provide clear ways to reset or clear the form Integration ■ Event Handling : Use the comprehensive event system for integration State Management : Integrate with your application's state management Error Handling : Handle authentication errors gracefully Loading States : Coordinate loading states with your backend Navigation : Handle post-authentication navigation Accessibility Guidelines ■ Semantic Structure ■ The component provides proper semantic structure: Form Elements : Proper form labels and associations Button Roles : Clear button roles and purposes Link Descriptions : Descriptive link text for screen readers Error Announcements : Clear error message announcements Focus Management : Logical focus order through form elements Screen Reader Support ■ Form Labels : Clear labels for all form inputs Error Messages : Proper announcement of validation errors Button Descriptions : Descriptive button labels Status

Updates : Clear status updates for loading and errors Navigation : Logical navigation through form elements Keyboard Navigation ■ Tab Order : Logical tab order through form elements Enter Key : Enter key for form submission Space Key : Space key for checkbox and button activation Focus Indicators : Clear focus indicators for keyboard users Shortcuts : Keyboard shortcuts for common actions Color and Contrast ■ WCAG Compliance : All text and interactive elements meet WCAG AA contrast ratios High Contrast Mode : Component works with system high contrast settings Color Independence : Information is not conveyed by color alone Visual Hierarchy : Clear visual distinction between form sections Responsive Behavior ■ Mobile Adaptations ■ The SSO login component automatically adapts to mobile screens: Touch Optimization : Optimized touch targets for mobile interaction Keyboard Handling : Proper mobile keyboard behavior Viewport Adaptation : Adapts to different mobile viewport sizes Form Layout : Responsive form layout for mobile devices Breakpoint Behavior ■ Desktop (>768px) : Full login interface with all features Mobile ($\leq 768\text{px}$) : Compact layout with optimized spacing Input Sizing : Responsive input field sizing Button Sizing : Appropriate button sizes for different screens Variant Considerations ■ XS (500px) : Compact login for embedded contexts SM (500px) : Small login for sidebar or modal contexts MD (500px) : Standard login for most applications LG (500px) : Large login for prominent authentication pages XL (560px) : Extra large login for enterprise applications Content Considerations ■ Form Layout : Form elements adapt to different screen widths Text Sizing : Appropriate text sizes for different variants Spacing : Optimized spacing for different screen sizes Touch Targets : Adequate touch target sizes for mobile Security Considerations ■ Input Sanitization ■ Username Validation : Validate username format and length Password Requirements : Implement strong password policies XSS Prevention : Sanitize all user inputs CSRF Protection : Implement CSRF tokens for form submission SQL Injection : Use parameterized queries on the backend Authentication Flow ■ Secure Transmission : Always use HTTPS for authentication Token Management : Implement secure token storage and rotation Session Security : Secure session management and timeout Audit Logging : Log authentication attempts and failures Rate Limiting : Prevent brute force attacks Data Protection ■ Password Hashing : Never store plain text passwords Personal Data : Minimize collection of personal information Data Retention : Implement appropriate data retention policies Privacy Compliance : Ensure compliance with privacy regulations Secure Storage : Use secure storage for sensitive data

```

import { Component } from "@angular/core";
import { AavaSSOLoginComponent, SSOLoginCredentials } from "@aava/play-core";

@Component({
  selector: "app-sso-login-example",
  standalone: true,
  imports: [AavaSSOLoginComponent],
  template: `
    <aava-sso-login
      [loading]="loading"
      [disabled]="disabled"
      [errorMessage]="errorMessage"
      variant="md"
      (login)="onLogin($event)"
      (ssoLogin)="onSSOLogin()"
      (forgotPassword)="onForgotPassword($event)"
      (troubleSignin)="onTroubleSignin()"
      (loginEvent)="onLoginEvent($event)"
    >
    </aava-sso-login>
  `,
})
export class SSOLoginExampleComponent {
  loading = false;
  disabled = false;
  errorMessage = "";

  onLogin(credentials: SSOLoginCredentials) {
    console.log("Login attempt:", credentials);
    this.loading = true;

    // Simulate authentication
    setTimeout(() => {
      this.loading = false;
      if (
        credentials.username === "admin" &&
        credentials.password === "password"
      ) {
        console.log("Login successful");
        this.errorMessage = "";
      } else {
        this.errorMessage = "Invalid username or password";
      }
    }, 2000);
  }

  onSSOLogin() {
    console.log("SSO login requested");
    // Redirect to SSO provider or open SSO modal
  }

  onForgotPassword(username: string) {
    console.log("Forgot password for:", username);
    // Open forgot password form or redirect
  }
}

```

```
onTroubleSignin() {
  console.log("Trouble signing in");
  // Open help modal or redirect to support
}

onLoginEvent(event: any) {
  console.log("Login event:", event);
  // Handle any login-related event
}
}
```